# Scrapper Code:

import requests

from bs4 import BeautifulSoup

import os

from concurrent.futures import ThreadPoolExecutor

import time

from urllib.parse import urljoin

import re

import json

import csv

from google.colab import files

import shutil

import threading

import glob

BASE\_URL = "https://papers.nips.cc"

DATASETS\_BENCHMARKS\_URL\_2021 = "https://datasets-benchmarks-proceedings.neurips.cc"

OUTPUT\_DIR = "/content/research\_papers\_scraped/"

METADATA\_DIR = "/content/metadata/"

THREAD\_COUNT = 50

MAX\_RETRIES = 3

TIMEOUT = 60

START\_YEAR = 1987

END\_YEAR = 2023

json\_lock = threading.Lock()

csv\_lock = threading.Lock()

def sanitize\_filename(filename):

return re.sub(r'[\\/\*?:"<>|]', '\_', filename)

def download\_pdf(session, pdf\_url, filename):

filename = sanitize\_filename(filename)

filepath = os.path.join(OUTPUT\_DIR, f"{filename}.pdf")

if os.path.exists(filepath):

print(f"File exists: {filepath}")

return

try:

with session.get(pdf\_url, stream=True, timeout=TIMEOUT) as response:

response.raise\_for\_status()

with open(filepath, 'wb') as f:

for chunk in response.iter\_content(chunk\_size=16384):

if chunk:

f.write(chunk)

print(f"Downloaded: {filepath}")

except Exception as e:

print(f"Failed to download {pdf\_url}: {e}")

def save\_metadata\_incremental\_json(metadata, year):

json\_file = f"neurips\_{year}.json"

with json\_lock:

with open(json\_file, "a", encoding="utf-8") as f:

f.write(json.dumps(metadata, indent=4) + "\n")

def save\_metadata\_incremental\_csv(metadata, year):

csv\_file = f"neurips\_{year}.csv"

headers = ["year", "title", "authors", "abstract", "pdf\_url", "paper\_url"]

row = [

metadata.get("year", ""),

metadata.get("title", ""),

"; ".join(metadata.get("authors", [])),

metadata.get("abstract", ""),

metadata.get("pdf\_url", ""),

metadata.get("paper\_url", "")

]

with csv\_lock:

write\_header = not os.path.exists(csv\_file)

with open(csv\_file, "a", newline="", encoding="utf-8") as f:

writer = csv.writer(f)

if write\_header:

writer.writerow(headers)

writer.writerow(row)

def process\_paper\_2022\_2023(session, paper\_url, year):

for attempt in range(MAX\_RETRIES):

try:

response = session.get(paper\_url, timeout=TIMEOUT)

soup = BeautifulSoup(response.text, 'html.parser')

title = soup.title.text.strip().replace(" - NeurIPS", "") if soup.title else "Untitled"

authors\_h4 = soup.find('h4', string=re.compile("Authors", re.I))

authors = []

if authors\_h4:

authors\_p = authors\_h4.find\_next('p')

if authors\_p:

authors = [a.strip() for a in authors\_p.get\_text(separator=",").split(",") if a.strip()]

abstract\_h4 = soup.find('h4', string=re.compile("Abstract", re.I))

abstract = "No abstract available"

if abstract\_h4:

abstract\_p = abstract\_h4.find\_next('p')

if abstract\_p:

abstract = abstract\_p.get\_text(strip=True)

pdf\_link = soup.find('a', href=lambda href: href and href.endswith('Paper-Conference.pdf'))

if not pdf\_link:

print(f"No PDF link found: {paper\_url}")

return

pdf\_url = urljoin(BASE\_URL, pdf\_link['href'])

download\_pdf(session, pdf\_url, f"{year}\_{title}")

metadata = {

"year": year,

"title": title,

"authors": authors,

"abstract": abstract,

"pdf\_url": pdf\_url,

"paper\_url": paper\_url

}

year\_dir = os.path.join(METADATA\_DIR, str(year))

os.makedirs(year\_dir, exist\_ok=True)

metadata\_filename = sanitize\_filename(f"{title}") + ".json"

metadata\_filepath = os.path.join(year\_dir, metadata\_filename)

with open(metadata\_filepath, 'w') as f:

json.dump(metadata, f, indent=4)

save\_metadata\_incremental\_json(metadata, year)

save\_metadata\_incremental\_csv(metadata, year)

return

except Exception as e:

print(f"Attempt {attempt+1} failed for {paper\_url}: {e}")

time.sleep(2 \*\* attempt)

print(f"Giving up: {paper\_url}")

def process\_paper(session, paper\_url, year):

for attempt in range(MAX\_RETRIES):

try:

response = session.get(paper\_url, timeout=TIMEOUT)

soup = BeautifulSoup(response.text, 'html.parser')

title\_tag = soup.find('h4')

title = title\_tag.get\_text(strip=True) if title\_tag else "Untitled"

pdf\_link = soup.find('a', href=lambda href: href and "Paper.pdf" in href)

if not pdf\_link:

print(f"No PDF link: {paper\_url}")

return

pdf\_url = urljoin(BASE\_URL, pdf\_link['href'])

authors\_h4 = soup.find('h4', string=re.compile("Authors", re.I))

authors = []

if authors\_h4:

authors\_p = authors\_h4.find\_next('p')

if authors\_p:

authors\_text = authors\_p.get\_text(strip=True)

authors = [a.strip() for a in authors\_text.split(",") if a.strip()]

abstract\_h4 = soup.find('h4', string=re.compile("Abstract", re.I))

abstract = "No abstract available"

if abstract\_h4:

abstract\_p = abstract\_h4.find\_next('p')

if abstract\_p and not abstract\_p.get\_text(strip=True):

abstract\_p = abstract\_p.find\_next('p')

if abstract\_p:

abstract = abstract\_p.get\_text(strip=True)

download\_pdf(session, pdf\_url, f"{year}\_{title}")

metadata = {

"year": year,

"title": title,

"authors": authors,

"abstract": abstract,

"pdf\_url": pdf\_url,

"paper\_url": paper\_url

}

year\_dir = os.path.join(METADATA\_DIR, str(year))

os.makedirs(year\_dir, exist\_ok=True)

metadata\_filename = sanitize\_filename(f"{title}") + ".json"

metadata\_filepath = os.path.join(year\_dir, metadata\_filename)

with open(metadata\_filepath, 'w') as f:

json.dump(metadata, f, indent=4)

save\_metadata\_incremental\_json(metadata, year)

save\_metadata\_incremental\_csv(metadata, year)

return

except Exception as e:

print(f"Attempt {attempt+1} failed for {paper\_url}: {e}")

time.sleep(2 \*\* attempt)

print(f"Giving up: {paper\_url}")

def process\_dataset\_benchmark\_papers(session, url):

try:

response = session.get(url)

soup = BeautifulSoup(response.text, 'html.parser')

paper\_links = soup.select('a[href$="Abstract.html"]')

for paper\_link in paper\_links:

paper\_url = urljoin(url, paper\_link['href'])

process\_paper(session, paper\_url, 2021)

except Exception as e:

print(f"Error processing datasets and benchmarks papers: {e}")

def get\_user\_years():

while True:

try:

user\_input = input(f"Enter up to 5 years (comma-separated) between {START\_YEAR} and {END\_YEAR}: ")

years = [int(year.strip()) for year in user\_input.split(",")]

if len(years) > 5:

print("Error: You can only select up to 5 years at a time.")

continue

invalid\_years = [year for year in years if year < START\_YEAR or year > END\_YEAR]

if invalid\_years:

print(f"Error: The following years are out of range: {invalid\_years}")

continue

return years

except ValueError:

print("Error: Please enter valid years (e.g., 1987,1990,2023).")

def main():

os.makedirs(OUTPUT\_DIR, exist\_ok=True)

os.makedirs(METADATA\_DIR, exist\_ok=True)

years\_to\_scrape = get\_user\_years()

print(f"Scraping years: {years\_to\_scrape}")

with requests.Session() as session:

response = session.get(BASE\_URL)

soup = BeautifulSoup(response.text, 'html.parser')

year\_links = soup.select('a[href^="/paper\_files/paper/"]')

with ThreadPoolExecutor(max\_workers=THREAD\_COUNT) as executor:

for year\_link in year\_links:

year\_url = urljoin(BASE\_URL, year\_link['href'])

try:

year = int(year\_url.split('/')[-1])

except (ValueError, IndexError):

print(f"Skipping invalid year URL: {year\_url}")

continue

if year in years\_to\_scrape:

print(f"Processing year: {year}")

try:

year\_response = session.get(year\_url)

year\_soup = BeautifulSoup(year\_response.text, 'html.parser')

if year > 2021:

paper\_links = year\_soup.select('a[href$="Abstract-Conference.html"]')

else:

paper\_links = year\_soup.select('a[href$="Abstract.html"]')

for paper\_link in paper\_links:

paper\_url = urljoin(year\_url, paper\_link['href'])

if year > 2021:

executor.submit(process\_paper\_2022\_2023, session, paper\_url, year)

else:

executor.submit(process\_paper, session, paper\_url, year)

except Exception as e:

print(f"Year {year\_url} error: {e}")

else:

print(f"Skipping year {year} (not selected)")

if 2021 in years\_to\_scrape:

executor.submit(process\_dataset\_benchmark\_papers, session, DATASETS\_BENCHMARKS\_URL\_2021)

if \_\_name\_\_ == "\_\_main\_\_":

main()

choice = input("Do you want to download the scraped data as a ZIP file to your local PC? (Enter 'L' for local download or 'C' for Colab cloud storage): ").strip().lower()

if choice == 'l':

FINAL\_OUTPUT\_DIR = "/content/final\_output"

os.makedirs(FINAL\_OUTPUT\_DIR, exist\_ok=True)

for f in glob.glob("/content/neurips\_\*.\*"):

if f.endswith(('.csv', '.json')):

shutil.copy(f, FINAL\_OUTPUT\_DIR)

shutil.copytree(METADATA\_DIR, os.path.join(FINAL\_OUTPUT\_DIR, "metadata"))

shutil.copytree(OUTPUT\_DIR, os.path.join(FINAL\_OUTPUT\_DIR, "research\_papers\_scraped"))

zip\_filename = "/content/neurips\_data.zip"

shutil.make\_archive("/content/neurips\_data", 'zip', FINAL\_OUTPUT\_DIR)

print(f"ZIP archive created: {zip\_filename}. Downloading now...")

files.download(zip\_filename)

shutil.rmtree(FINAL\_OUTPUT\_DIR)

else:

print("Files remain in the Colab cloud. You can access them in the workspace.")

# Automation Code:

from google import genai

import pandas as pd

import time

GEMINI\_API\_KEY = "AIzaSyC9-pO3MlbA5063BXgkaifKeYbuqsq4sSQ"

client = genai.Client(api\_key=GEMINI\_API\_KEY)

LABELS = ["Deep Learning", "Computer Vision", "Reinforcement Learning", "NLP", "Optimization"]

def classify\_paper(title, abstract):

prompt = (

f"Classify the following research paper into one of these categories: {', '.join(LABELS)}. "

"If it does not clearly fit into any of these, reply with 'Other'.\n\n"

f"Title: {title}\n\nAbstract: {abstract}\n\nCategory:"

)

try:

response = client.models.generate\_content(

model="gemini-2.0-flash",

contents=prompt

)

classification = response.text.strip()

print(f"Assigned category: {classification}")

return classification

except Exception as e:

print(f"Gemini API error: {e}")

return "Error"

def annotate\_dataset(input\_csv, output\_csv):

try:

df = pd.read\_csv(input\_csv)

except Exception as e:

print(f"Error reading {input\_csv}: {e}")

return

if "title" not in df.columns or "abstract" not in df.columns:

print("Error: The CSV file must have 'title' and 'abstract' columns.")

return

annotations = []

for idx, row in df.iterrows():

title = row["title"]

abstract = row["abstract"]

print(f"\nAnnotating paper {idx + 1}/{len(df)}: {title}")

label = classify\_paper(title, abstract)

annotations.append(label)

time.sleep(2)

df["Category"] = annotations

try:

df.to\_csv(output\_csv, index=False)

print(f"\nAnnotated dataset saved to: {output\_csv}")

except Exception as e:

print(f"Error saving annotated dataset: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

input\_csv\_file = "C:/Users/asfan/Downloads/neurips\_data (2)/neurips\_1999.csv"

output\_csv\_file = "C:/Users/asfan/Downloads/neurips\_data (2)/neurips\_papers\_annotated.csv"

annotate\_dataset(input\_csv\_file, output\_csv\_file)